1. Problem Definition
    1. The context
        - With busy lives people have less time to discover their artistic preferences.
        - Advanced technology made it possible to discover and consume new content and customers' musical taste.
        - As most of the internet-based companies revenue depends on the time consumers spend on their platforms, it becomes important for those companies to determine what kind of content will provide more customer time spent on their website and better customer experience.
        - **Biggest challenge** – figure out content that will be consumed by the customers.
        - Spotify is an example of such company that thanks to its music recommendations (smart recommendation system – ability to recommend the best next song to each and every customer based on their preferences) was able grow significantly in the market.

    2. The objectives
        - The goal is to build a recommendation system to propose the top 10 songs for a user based on the likelihood of listening to those songs.

    3. The key questions - What are the key questions that need to be answered?

        - Which songs are the most popular?
        - What songs are users most likely listen to?

        - How is the data prepared?
        - Which techniques should be used?
        - How would the models be evaluated and how to choose the best one?
        - How are the hyper parameters tuned?

    4. The problem formulation - What is it that we are trying to solve using data science?

        - We're trying to find the best recommendations of songs for users using recommendation systems (3 different models).

2. Data Exploration
    1. Data Description
        - The core data is the Taste Profile Subset released by the Echo Nest as part of the Million Song Dataset.
        - There are two files in this dataset.
            - The first file contains the details about the song id, titles, release, artist name, and the year of release.
            - The second file contains the *user_id*, *song_id*, and the *play_count* of users, so info about songs and users' interactions.
        - *song_data*
            - *song_id* - A unique id given to every song
            - *title* - Title of the song
            - *Release* - Name of the released album
            - *Artist_name* - Name of the artist
            - *year* - Year of release

- *count_data*
    - *user _id* - A unique id given to the user
    - *song_id* - A unique id given to the song
    - *play_count* - Number of times the song was played
- The common column in both data sets is *song_id* and that's what we're going to use to merge those two datasets

2. Observations & Insights
    - Most data types are object (besides columns *Unnamed*, *play_count* and *year*).
    - No missing data in the *count_df* dataset.
    - Some missing data in *song_df* dataset – 15 missing records for *title* and 5 missing records for *release.*
    - Year in *song_df* has a lot values of 0 (484,424 compared to 39,4141 for 2007) so those records will be later dropped.
    - We're merging two data sets on the *song_id* column.
    - *User_id* and *song_id* are encrypted and will be encoded to numeric features (so we can use it in the model).
    - *Title*, *Release* and *artist_name* could be changed into string type (from object).
    - Data contains users who listened to very little songs and also songs that have been listened only few times, we're going to drop those records, saving only users who listened to at **least 90 or more** songs and songs that have been listened by **120 or more** users in our final data frame.
    - Final data frame has **3,155** unique users, **563** unique songs and **232** unique artists.
    - We have 3,155 * 563 = **1,776,265** possible interactions between users and songs.
    - We already have 117,876 interactions in our final data frame, so we have 1,776,265 - 117,876 = **1,658,389** possible interactions left.

3. Proposed approach

    1. Potential techniques

    - In this study, I'll build three types of recommendation systems:

        - **Knowledge/Rank Based recommendation system**
        - **Similarity-Based Collaborative filtering (user-user, item-item models)**
        - **Matrix Factorization Based Collaborative Filtering**

    2. Overall solution design

    - Loading data and Exploratory Data Analysis

        1. Checking for missing values.
        2. Summary statistics.
        3. Checking the number of unique users and songs.
        4. Data preparation.

- Create final data frame: merge two data frames and also chose only users who listened to at least 90 songs and songs which have been listened to by at least 120 users also dropping records with *play_count* greater than 5.
- Create models
    - **Model 1: Rank Based Recommendation System**
        - Create function top_10_songswhich will recommend top 10 songs for a user based on the likelihood of listening to those songs.
    - **Model 2: Collaborative Filtering Recommendation System**
        - **User-user** and **item -item**

            - Building a baseline user-user and item-item similarity-based recommendation systems.
                - Initialize the KNNBasic model using sim_options provided, Verbose=False, and setting random_state=1.
                - Fit the model on the training data.
                - Use the precision_recall_at_k function to calculate the metrics on the test data.
                - Improving similarity-based recommendation system by tuning its hyperparameters.
                - Create function top_10_songswhich will recommend top 10 songs for a user based on the likelihood of listening to those songs.

    - **Model 3: Matrix factorization**

        - Singular Value Decomposition (SVD).

        - Fit the SVD model using the hyperparameters from GridSearchCV.

        - Get recommendations.

3. Measures of success - What are the key measures of success to compare potential techniques?

    - **For performance evaluation** of above-mentioned models' **precision@k and recall@k** will be used.
    - Using these two metrics, the **F_1 score** will be calculated for each working model.
    - **RMSE** (Root Mean Square Error) will also be used to evaluate best solution.
    - Best model will have greatest **F-1** (and ideally **Precision and Recall**) and minimum **RMSE.**